IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appellant:	EZZAT	Patent Application	
Application No.:	10/769,594	Group Art Unit:	2135
Filed:	January 30, 2004	Examiner:	Gyorfi, Thomas A.

For: PROVIDING A FLEXIBLE PROTECTION MODEL IN A COMPUTER SYSTEM BY DECOUPLING PROTECTION FROM COMPUTER PRIVILEGE LEVEL

APPEAL BRIEF

Table of Contents

I. Real Party in Interest

The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 11445 Compaq Center Drive West, Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

II. Related Appeals and Interferences

There are no related appeals or interferences known to the Appellant.

III. Status of Claims

Claims 1 and 3-26 are pending. Claim 2 has been cancelled. Claims 1 and 3-26 are rejected. This Appeal involves Claims 1 and 3-26.

IV. Status of Amendments

All proposed amendments have been entered.

V. Summary of Claimed Subject Matter

Independent Claim 1 recites, "A method 300 of providing flexible protection in a computer system by decoupling protection from privilege (240B, 250B)." This embodiment is described at least at paragraph 0028 on page 7; paragraphs 0030-0045 on pages 8-11; Figures 2A, 2B; and Figures 3A, 3B. "Enabling receipt of information describing two or more types of protection," is described at least at Figure 2A; step 302, Figure 3A; paragraphs 0072-0075 on pages 16 and 17; lines 6-9 of paragraph 0063 on page 15; and paragraph 0066 on page 15. "Wherein said portions of code are not required to be associated with one or more object oriented classes," is described, according to various embodiments, at least at paragraphs 0031, 0033, 0035, 0076, 0077, and 0094. "Enabling receipt of information describing a relationship between said two or more types of protection and portions of code C1B-C9B that are executed in a same privilege level 250B of the computer system 200B, wherein said relationship is not required to be linear," is described at least at Figure 2A; Figure 2B; step 302, Figure 3A; and paragraph 0072 on page 16. "Enabling the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code C1B-C9B, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code," is described at least at Figure 2A; steps 306-320, Figures 3A, 3B; paragraphs 0076-0082 on pages 17 and 19; and last 4 lines of paragraph 0053 on page 13.

Independent Claim 10 recites, "A method 300 of providing flexible protection in a computer system by decoupling protection from privilege (240B, 250B)." This embodiment is described at least at paragraph 0028 on page 7; paragraphs 0030-0045 on pages 8-11; Figures 2A, 2B; and Figures 3A, 3B. "Detecting a request from a first portion of code to access a second portion of code, wherein said first and second portions of code are executed in a same privilege level 250B of said computer system 200B," is described at least at steps 308, 310, Figure 3A; and paragraph 0078-0079 on page 18. "Wherein said portions of code are not required to be associated with one or more object oriented classes," is described, according to various embodiments, at least at paragraphs 0031, 0033, 0035, 0076, 0077, and 0094. "Determining whether said first portion of code is allowed to access said second portion of code based on information describing two or more types of protection and also based on information describing a relationship between said two or more types of protection and said portions of code, wherein said relationship is not required to be linear," is described at least at step 312, Figure 3B; lines 6-9 of paragraph 0063 on page 15; and paragraph 0080 on page 18. "If said relationship specifies that said first portion of code may access said second portion of code allowing said first portion of code to access said second portion of code," is described at least at step 316, Figure 3B; and paragraph 0081 on page 18. "Else not allowing said first portion of code to access said second portion of code," is described at least at step 320, Figure 3B; and paragraph 0082 on page 19.

Independent Claim 12 recites, "A computer system 200A." This embodiment is described at least at Figure 2A; and paragraphs 0030-0037 on pages 8 to 10. "A memory unit 402" is described at least at Figure 4; and paragraphs 0091-0093 on page 22. "A processor 401 coupled to the memory unit 402, the processor 401 for executing a method 300 for enforcing protection in a computer system by decoupling protection from privilege 240B, 250B," is described at least at Figure 2A; paragraphs 0030-0037 on pages 8 to 10; Figure 4; and paragraphs 0091-0093 on page 22. "Enabling at a user interface 240A receipt of information describing two or more types of protection," is described at least at paragraph 0033 on pages 8-9; step 302, Figure 3A; paragraphs 0072-0075 on pages 16 and 17; lines 6-9 of paragraph 0063 on page 15; paragraph 0066 on page 15; and Figure 2A. "Enabling at the user interface 240A receipt of information describing a relationship between said two or more types of protection and portions of code are executed in a same privilege level 250B of the computer system 200B, wherein said relationship is not required to be linear," is described at least at paragraph 0033 on pages 8-9; Figure 2A; Figure 2B; step 302, Figure 3A; and paragraph 0072 on page 16. "Wherein said portions of code are not required to be associated with one or more object oriented classes," is described, according to various embodiments, at least at paragraphs 0031, 0033, 0035, 0076, 0077, and 0094. "Enabling at a link-editor 220A the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code," is described at least at Figure 2A; paragraphs 0036-0037 on pages 9-10; steps 306-320, Figures 3A, 3B; paragraphs 0076-0082 on pages 17 and 19; and last 4 lines of paragraph 0053 on page 13.

Independent Claim 15 recites, "A computer system 200B." This embodiment is described at least at Figure 2B; and paragraphs 0038-0045 on pages 10-11. "A memory unit 402" is described at least at Figure 4; and paragraphs 0091-0093 on page 22. "A processor 401 coupled to the memory unit 402, the processor 401 for executing a method 300 for providing flexible protection in a computer system 200B by decoupling protection from privilege 240B, 250B" is described at least at Figures 2A, 2B; paragraphs 0038-0045 on pages 10-11; Figure 4; paragraphs 0091-0093 on page 22; paragraph 0028 on page 7; paragraphs 0030-0045 on pages 8-11; and Figures 3A, 3B. "Detecting at a memory manager 260B a request from a first portion of code to access a second portion of code, wherein said first and second portions of code are executed in a same privilege level 250B of said computer system 200B," is described at least at Figure 2B; paragraphs 0041-0042 on pages 10-11; steps 308, 310, Figure 3A; and paragraphs 0078-0079 on page 18. "Wherein said portions of code are not required to be associated with one or more object oriented classes," is described, according to various embodiments, at least at paragraphs 0031, 0033, 0035, 0076, 0077, and 0094. "Determining at said memory manager 260B whether said first portion of code is allowed to access said second portion of code based on information describing two or more types of protection and also based on information describing a relationship between said two or more types of protection and said portions of code, wherein said relationship is not required to be linear," is described at least at paragraph 0042 on pages 10-11; step 312, Figure 3B; lines 6-9 of paragraph 0063 on page 15; and paragraph 0080 on page 18. "If said relationship specifies that said first portion of code may access said second portion of code, then allowing at said memory manager said first portion of code to access said second portion of code," is described at least at step 316, Figure 3B; and paragraph 0081 on page 18. "Else not allowing at said memory manager said first portion of code to access said second portion of code," is described at least at step 320, Figure 3B; and paragraph 0082 on page 19.

Independent Claim 17 recites, "A computer-usable medium having computer-readable program code embodied therein for causing a computer system to perform a method 300 of providing flexible protection in a computer system 200B by decoupling protection from privilege (240B, 250B)." This embodiment is described at least at paragraph 0028 on page 7; paragraphs 0030-0045 on pages 8-11; Figures 2A, 2B; and Figures 3A, 3B. "Enabling receipt of information describing two or more types of protection," is described at least at Figure 2A; step 302, Figure 3A; paragraphs 0072-0075 on pages 16 and 17; lines 6-9 of paragraph 0063 on page 15; and paragraph 0066 on page 15. "Wherein said portions of code are not required to be associated with one or more object oriented classes," is described, according to various embodiments, at least at paragraphs 0031, 0033, 0035, 0076, 0077, and 0094. "Enabling receipt of information describing a relationship between said two or more types of protection and portions of code C1B-C9B that are executed in a same privilege level 250B of the computer system 200B, wherein said relationship is not required to be linear," is described at least at Figure 2A; Figure 2B; step 302, Figure 3A; and paragraph 0072 on page 16. "Enabling the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code C1B-C9B, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code," is described at least at Figure 2A; steps 306-320, Figures 3A, 3B; paragraphs 0076-0082 on pages 17 and 19; and last 4 lines of paragraph 0053 on page 13.

VI. Grounds of Rejection to Be Reviewed on Appeal

1. Claims 17-25 stand rejected under 35 U.S.C. 101.

2. Claims 1 and 3-26 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,125,447 by Gong (referred to herein as "Gong") in view of "The C Book—Structures" (referred to herein as "C Book—Structures").

VII. Argument

1. Whether Claims 17-25 Recite Statutory Subject Matter Under 35 U.S.C. 101

Claims 17-25 stand rejected under 35 U.S.C. 101. On January 26, 2010, the USPTO published guidance on the Subject Matter Eligibility of Computer Readable Media, which recites in part (emphasis added):

> The United States Patent and Trademark Office (USPTO) is obliged to give claims their broadest reasonable interpretation consistent with the specification during proceedings before the USPTO. *See In re Zletz,* 893 F.2d 319 (Fed. Cir. 1989) (during patent examination the pending claims must be interpreted as broadly as their terms reasonably allow). The broadest reasonable interpretation of a claim drawn to a computer readable medium (also called machine readable medium and other such variations) typically covers forms of non-transitory tangible media and transitory propagating signals *per se* in view of the ordinary and customary meaning of computer readable media, particularly when the specification is silent. *See* MPEP 2111.01. When the broadest reasonable interpretation of a claim covers a signal *per se*, the claim must be rejected under 35 U.S.C. § 101 as covering non-statutory subject matter. *See In re Nuijten,* 500 F.3d 1346, 1356-57 (Fed. Cir. 2007) (transitory embodiments are not directed to statutory subject matter) and *Interim Examination Instructions for Evaluating Subject Matter Eligibility Under 35 U.S.C. § 101*, Aug. 24, 2009; p. 2.

In particular, Appellant respectfully notes that the guidance specifically requires that claims be given "their broadest reasonable interpretation consistent with the specification" (emphasis added).

Appellant notes that the instant Office Action rejects Claims 17-25 under the assertion that it is within the scope of the disclosure that Claims 17-25 are directed toward a transitory propagating signal *per se*, and are thus non-statutory. However, Appellant respectfully submits that the specification describes only statutory embodiments of a computer readable medium and is silent with respect to any non-statutory embodiments of a computer readable medium. Therefore, Appellant respectfully submits that Claims 17-25, when reasonably interpreted consistent with the specification, are directed toward statutory subject matter, and thus overcome the instant rejection under 35 U.S.C. § 101.

The Office Action quoted paragraph 0070 of the instant Application. The Office Action states on page 3 lines 4-5, "Examiner utterly fails to see how this one recitation could possibly be read to exclude signal embodiments, as alleged by the Applicant." Appellant respectfully submits that Appellant stated "Applicant respectfully submits that the specification describes only statutory embodiments of a computer readable medium and is silent with respect to any non-statutory embodiments of a computer readable medium" (emphasis added). Further, Appellant respectfully reiterates that the guidance specifically requires that claims be given "their broadest reasonable interpretation consistent with the specification" (emphasis added).

Therefore, Appellant respectfully submits that Claims 17-25, when reasonably interpreted consistent with the specification, are directed toward statutory subject matter, and thus overcome the instant rejection under 35 U.S.C. § 101.

2. Whether Claims 1 and 3-26 Are Patentable Over Gong In View of C Book—Structures Under 35 U.S.C. 103.

Appellant has reviewed the asserted art and respectfully submits that the asserted art does not teach or suggest Claims 1 and 3-26 for at least the following reasons.

"As reiterated by the Supreme Court in KSR, the framework for the objective analysis for determining obviousness under 35 U.S.C. 103 is stated in Graham v. John Deere Co., 383 U.S. 1, 148 USPQ 459 (1966). Obviousness is a question of law based on underlying factual inquiries" including "[a]scertaining the differences between the claimed invention and the prior art" (MPEP 2141(II)). "In determining the differences between the prior art and the claims, the question under 35 U.S.C. 103 is not whether the differences themselves would have been obvious, but whether the claimed invention as a whole would have been obvious" (emphasis in original; MPEP 2141.02(I)).

Appellant notes that "[t]he prior art reference (or references when combined) need not teach or suggest all the claim limitations, however, Office personnel must explain why the difference(s) between the prior art and the claimed invention would have been obvious to one of ordinary skill in the art" (emphasis added; MPEP 2141(III)).

Appellant respectfully submits that "[i]t is improper to combine references where the references teach away from their combination" (emphasis added; MPEP 2145(X)(D)(2); *In re Grasselli*, 713 F.2d 731, 743, 218 USPQ 769, 779 (Fed. Cir. 1983)). Appellant respectfully notes that "[a] prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention" (emphasis in original; MPEP 2141.02(VI); *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 (1984)). Further, "[a] reference will teach away if it suggests that the line of development flowing from the reference's disclosures is unlikely to be productive of the result sought by the applicant. *In re Gurley*, 31 USPQ2d 1130 (Fed. Cir. 1994)."

Appellant respectfully submits that Gong does not teach or suggest "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by independent Claim 1, as will become more evident. Further, Appellant respectfully submits that Gong teaches away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by independent Claim 1, as will become more evident.

For example, Gong states at Col. 1 lines 60-62, "Recently developed methods of running applications involve the automatic and immediate execution of software code loaded from remote sources over the network." Gong further states at Col. 2 lines 44-50,

> Based on the foregoing, it is clearly desirable to develop a method which reduces the effort and in-depth knowledge required to modify permissions established for the sources of code being executed by a computer system. It is further desirable to develop a method which reduces the effort and in-depth knowledge required to create new permissions.

Accordingly, Appellant understands Gong's intended purpose is to reduce the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein). As will become more evident, Appellant understands Gong's code to be instances of OO classes and to use instances of OO classes to achieve Gong's intended purposes of reducing the effort and in-depth knowledge required to modify permissions and to create new permissions to provide security for the code.

Appellant respectfully submits that Gong achieves Gong's intended purpose by mapping OO classes to protection domains (see Gong abstract; Col. 2 line 66-Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein). For example, Gong states in the last sentence of the abstract, "When an object requests an action, a determination is made as to whether the action is permitted based on the class to which the object belongs and the association between classes and protection domains." Gong states at Col. 2 line 66 to Col. 3 line 1, "For example, based on policy data, an association between Class CA and protection domain PA, and class CB and protection domain PB is established." Gong states at Col. 6 lines 44-45, "...without requiring specialized knowledge of complex security-management techniques." Gong states at Col. 6 lines 55-62,

> Code executor 210 executes code which code executor 210 receives from code stream 220. One example of a code executor is a Java virtual machine. A Java virtual machine interprets code called byte code. Byte code is code generated by a Java compiler from source files containing text. The Java virtual machine is described in detail in Tim Lindholm & Frank Yellin, *The Java Virtual Machine Specification* (1996).

Further, Gong states at Col. 8 lines 39-43, "According to an embodiment of the present invention, protection domains are used to enforce security within a computer system. A protection domain can be viewed as a set of permissions granted to one or more principles." Gong states at Col. 10 lines 49-51, "Then protection domain object 282 is created and populated with the permission just mentioned." Gong states at Col. 10 lines 56-57, "Next, in step 428, the mapping of the class to the protection domain is established." Gong states at Col. 10 lines 63-64, "Creating an association of classes to protection domains in the manner just described offers several advantages." Gong states at Col. 12 lines 39-47 (emphasis added),

> Examining the permissions of a particular protection domain associated with an object begins by determining an object's class. <u>A code executor, such as a Java Virtual machine, provides that each object incorporates a method which returns the class of an object</u>. In this example, the first object with a method on the call stack is object a. Access controller 280 invokes the method that returns the class of object a.

Accordingly, Appellant understands Gong to require his code to be instances of object oriented (OO) classes so that Gong can easily map Gong's protection domain, which represent Gong's protection attributes, to the OO classes (see Gong abstract; Col. 2 line 66- Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein).

By mapping OO classes to protection domains (see Gong abstract; Col. 2 line 66- Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein), Appellant understands Gong to achieve Gong's intended purpose of reducing the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code (e.g., the instances of the OO classes) that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein). Therefore, Appellant understands Gong to <u>require</u> his code to be instances of OO classes in order to achieve Gong's intended purpose.

In contrast, Claim 1 recites "wherein said portions of code are not required to be associated with one or more object oriented classes." Appellant respectfully submits that requiring code to be instances of OO classes teaches away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by Claim 1. Therefore, Appellant respectfully submits that Claim 1 is patentable over Gong.

Further, since Appellant understands Gong to teach away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by Claim 1, Appellant respectfully submits that there is no motivation to combine Gong with any other asserted art, such as C Book—Structures. Therefore, Appellant respectfully submits that Gong does not teach or suggest Claim 1 and even teaches away from Claim 1.

Appellant respectfully submits that C-Book Structures does not remedy the deficiencies in Gong. For example, the title of C-Book—Structures is "The C Book—Structures." As any one of ordinary skill in the art understands, C is a purely non-object oriented language. However, as discussed herein, Appellant understands Gong to <u>require</u> his code to be instances of OO classes. Since Gong requires his code to be instances of OO classes, Appellant respectfully submit that modifying Gong to use a purely non-object oriented language would render Gong in operable for Gong's intended purpose. Therefore, Appellant respectfully submits that there is no motivation to combine Gong and C Book—Structures because Appellant understands Gong and C Book—Structures to teach away from each other. Further, Appellant respectfully submits that there is no motivation to combine Gong with any other asserted art, such as C Book—Structures, because as discussed herein, Appellant understands Gong to teach away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by Claim 1.

OFFICE ACTION'S ASSERTIONS

The Office Action states on page 6 line 20 to page 7 line 4,

> Gong's disclosure is limited to explicitly discussing the preferred embodiment wherein all the pertinent software is implemented as Java objects, Java being a well-known object-oriented programming language with classes (col. 6, line 45 – col. 7, line 60). However, Gong merely assumes that the object oriented requirement is true (Ibid, particular col. 6, lines 65-66); yet his preferred embodiment is illustrative but not restrictive, and variations as to the specifics of how his invention is implemented are permitted (col. 13, lines 23-30).

Appellant respectfully agrees with the Office Action's statement on page 6 lines 20-21, "Gong's disclosure is limited to explicitly discussing…all the pertinent software is implemented as Java objects."

Appellant respectfully disagrees that Gong is merely discussing a preferred embodiment.

First, Appellant shall discuss the portions of Gong (e.g., Col. 6 line 45 to Col. 7 line 60, Col. 6; lines 65-66, Col. 13 lines 23-30) that the Office Action referred to on page 4 lines 3-9 of the Office Action.

The Office Action appears to assert on page 4 lines 4-6 that Gong indicates that Java is a well known object-oriented programming language with classes at Col. 6 line 45 to Col. 7 line 60. Appellant agrees that Gong indicates that Java is a well known object-oriented programming language with classes.

At Col. 6 lines 65-66, Gong states, "Consequently, the code is in the form of methods associated with objects that belong to classes." Accordingly, Appellant understands Gong to use object oriented classes to divide his code into portions that security can be provided to, as discussed herein.

Gong states at Col. 13 lines 23-30,

> In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Appellant respectfully notes that Gong does not describe any "variations as to the specifics" at Col. 13 lines 23-30. Instead, Appellant understands Gong to provide boiler plate language indicating that modifications and changes may be made without departing from the broader spirit and scope of the invention at Col. 13 lines 23-30. Appellant understands Gong to describe all of Gong's embodiments in the context of object oriented classes and understands Gong to be silent with respect using non-object oriented languages for any of Gong's embodiments.

Further, as discussed herein, by mapping OO classes to protection domains (see Gong abstract; Col. 2 line 66- Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein), Appellant understands Gong to achieve Gong's intended purpose of reducing the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code (e.g., the instances of the OO classes) that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein).

Therefore, Appellant respectfully disagrees that Gong is merely discussing a preferred embodiment. Instead, Appellant understands Gong to require his code that he provides security for to be instances of OO classes.

In yet another example, Gong states at Col. 2 lines 44-64 (emphasis added),

> Based on the foregoing, it is clearly desirable to develop a method which reduces the effort and in-depth knowledge required to modify permissions established for the sources of code being executed by a computer system. It is further desirable to develop a method which reduces the effort and in-depth knowledge required to create new permissions.
>
> SUMMARY OF THE INVENTION
>
> A method and system are provided for implementing security policies within a computer system. The security mechanism makes use of structures referred to herein as "protection domains" to organize, represent and maintain the security policies that apply to the computer system.
>
> According to one aspect of the invention, protection domains are established based on policy data, where each protection domain is associated with zero or more permissions. An association is established between the protection domains and classes of objects (i.e., instantiations of the classes) that may be invoked by the computer system. When an object requests an action, a determination is made as to whether the action is permitted for that object…

Accordingly, Appellant understands Gong to state at Col. 2 lines 44-50 that his intended purpose is to reduce the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code (e.g., the instances of the OO classes) that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein). Further, Appellant notes that immediately after Gong states his intended purpose at Col. 2 lines 44-50 quoted herein, Gong teaches establishing an association between protection domains and classes of objects (i.e. instantiations of the classes) that may be invoked by the computer system (see Gong Col. 2 lines 60-63 quoted herein). Accordingly, Appellant understands Gong to require his code to be instances of OO classes in order to achieve his intended purpose. Therefore, Appellant understands Gong to teach away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by independent Claim 1.

Accordingly, Appellant respectfully submits that Claim 1 is patentable over the Gong C Book—Structure combination.

RESPONSE TO ARGUMENTS

The Office Action states on page 3 lines 12-17, "Appellant has naively assumed that in order for one to have the ability to load code from a remote resource on a network, said code must be an instance of an object-oriented [OO] class (see the amendment of 4/28/11, page 11, 5th paragraph; page 15, last paragraph; page 16, last paragraph; etc)."

Appellant respectfully submits that the Office Action's statement on page 3 lines 12-17 is a mischaracterization of Appellant's statements. To reiterate, Appellant understands Gong's intended purpose is to reduce the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein). Appellant understands Gong to require his code to be instances of object oriented (OO) classes so that Gong can easily map Gong's protection domains, which represent

Gong's protection attributes, to the OO classes (see Gong abstract, Col. 2 lines 66-Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein). By mapping OO classes to protection domains (see Gong abstract; Col. 2 line 66-Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57; Col. 12 lines 39-47 quoted herein), Appellant understands Gong to achieve Gong's intended purpose of reducing the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code (e.g., the instances of the OO classes) that, for example, may be loaded from remote resources over a network (see Gong Col. 2 lines 44-50; Col. 1 lines 60-62 quoted herein). Therefore, Appellant understands Gong to require his code to be instances of OO classes. Appellant respectfully submits that Gong's requiring his code to be instances of OO classes in order to achieve Gong's intended purpose does not mean that all code that is remotely loaded, for example, on the Internet is object oriented.

The Office Action goes on to state on page 7 lines 17-20, "This is incorrect, as those of ordinary skill in the art were fully aware of alternative means to achieve the equivalent effect; one such non-limiting example would be through the use of Remote Procedure Calls (RPCs), as evidenced by the Marshall reference enclosed herein." Appellant respectfully submits that although the prior art teaches that remote procedure calls can be made to programs written in non-object oriented languages this is not evidence that Gong envisioned a way to provide security when remotely invoking non-object oriented programs. Further, if it was so common for non-object oriented programs to be remotely invoked, then it was all the more important for Gong to recite and enable embodiments covering the remote invocation of non-object oriented programs. However, the Office Action itself admits on page 6 lines 20-21 that, "Gong's disclosure is limited to explicitly discussing…all the pertinent software is implemented as Java objects."

Gong states on Col. 11 lines 40-44, "A code executor, such as a Java virtual machine, maintains for each thread or process a call stack of the object methods invoked by the thread or process. The call stack reflects the calling hierarchy between the methods that have been invoked but not yet completed by the thread or process." Gong further states at Col. 11 lines 55-57, "Note that objects corresponding to the method invocations in the call stack are each associated with a protection domain." Gong further states at Col. 12 line 4-11 (emphasis added), "In other words, <u>a method associated with object a invoked a method that may perform the particular action.</u> Object a is requesting to write to <u>file '/tmp/temporary.'</u> The permission required to perform this action is a 'write' permission for file '/tmp/temporary'. The permission is required to perform a requested action is herein referred to as a required permission."

Accordingly, Appellant understands Gong to teach that a method, such as x, associated with object "a" invokes another method, which shall be referred to as "m," to access data, which shall be referred to as "d," where "m" and "d" are associated with another object, which shall be referred to as "A" (see Gong Col. 11 lines 40-44, Col. 11 lines 55-57, Col. 12 lines 4-11 quoted herein). Appellant understands that in the illustration described by Gong at Col. 11 line 34 on, the data "d" is the file "tmp/temporary" (see Gong Col. 12 line 8 quoted herein).

As discussed herein, Appellant understands Gong to teach mapping protection domains to OO classes (see Gong abstract; Col. 2 line 66-Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57; Col. 12 lines 39-47 quoted herein). Further, Appellant understands Gong to teach using a call stack to keep track of which OO methods associated with the OO classes are the invokers and the invokees (see Gong Col. 11 lines 40-44, Col. 11 lines 55-57 quoted herein) in order to determine if the invoker a.x has a required permission for accessing data "d" using a method "m" is available to the invoker a.x to access the data "d" (see Gong Col. 12 lines 4-11 quoted herein). Further, Appellant understands Gong to rely on the object oriented function where each object incorporates a method that returns the class that the object is an instance of (see Gong Col. 12 lines 39-47 quoted herein). Accordingly, Appellant understands Gong to teach that when an invoker a.x of a method A.m to access data "d" is on the top of the stack (see Gong Col. 11 lines 40-44, Col. 11 lines 55-57 quoted herein), the objects a and A can return the classes that they are associated with (see Gong Col. 12 lines 39-47 quoted herein), then the returned classes can be used to determine the protection domain that is required to enable a.x to access data "d" using method A.m (see Gong abstract; Col. 2 line 66-Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57; Col. 11 line 65 to Col. 12 line 11; Col. 12 lines 39-47; Figure 6 quoted herein).

The Office Action states at page 7 lines 4-6, "In that vein, those of ordinary skill in the art would have known that other programming languages predating the object-oriented programming phenomenon nevertheless allowed for data objects and methods to manipulate them." First, Appellant respectfully submits that OO objects include data and methods for accessing that data. Second, Appellant respectfully submits that programming languages predating OO did not provide for OO classes and instances of those classes. In object oriented programming, an OO instance "A" includes data "d" and a method "m" for accessing that data "d." A method a.x will invoke the method A.m in order to access the data "d". Further, another method, such as e.f would also invoke the same method A.m in order to access the data "d."

In contrast, in non-objected oriented programming, a program would access the data itself without invoking a method. Further in non-objected oriented programming, several non-object oriented programs can access the same data without invoking a method. For example, <u>two programs C and D could directly access the same data without invoking a method in non-object oriented programming</u>. This is one of the reasons that Appellant respectfully submits that Gong's principles of operation, which rely on object oriented programming and a call stack that tracks the hierarchical order of OO methods and associating protection domains with the respective OO classes teaches away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by Claim 1.

Further, there are other reasons that Appellant understands Gong to teach away from "wherein said portions of code are not required to be associated with one or more object oriented classes," as recited by Claim 1.

First, as the Office Action admits, non-object oriented programs can be loaded remotely (see instant Office Action page 3 lines 11-20).

Second, as the Office Action admits on page 6 lines 20-21, "Gong's disclosure is limited to explicitly discussing…all the pertinent software is implemented as Java objects." Accordingly, Appellant understands Gong to be silent with respect to teaching embodiments where non-object oriented programs are loaded remotely despite the fact that Gong would have known that non-object oriented programs can be loaded remotely.

Third, Appellant respectfully submits that Gong relies on object oriented programming and a call stack that tracks the hierarchical order of OO methods (see Gong Col. 11 lines 40-44, Col. 11 lines 55-57 quoted herein), using the OO feature where each object returns the class that the object is an instance of (see Gong Col. 12 lines 39-47 quoted herein) and associating protection domains with OO classes (see Gong abstract; Col. 2 line 66- Col. 3 line 1; Col. 6 lines 44-45; Col. 6 lines 55-62; Col. 8 lines 39-43; Col. 10 lines 49-51; Col. 10 lines 56-57, Col. 12 lines 39-47 quoted herein) in order to achieve Gong's intended purpose of reducing the effort and in-depth knowledge required to modify permissions or to create new permissions to provide security for the code that (see Gong Col. 2 lines 44-50, Col. 1 lines 60-62 quoted herein).

SUMMARY

For at least these reasons, Appellant respectfully submits that Claim 1 is patentable over the Gong C Book—Structures combination. For similar reasons, Appellant respectfully submits that independent Claims 10, 12, 15 and 17 are patentable over the Gong C Book—Structures combination in that independent Claims 10, 12, 15 and 17 also recite "wherein said portions of code are not required to be associated with one or more object oriented classes."

Claims 3-9 depend on independent Claim 1. Claim 11 depends on independent Claim 12. Claims 13 and 14 depend on independent Claim 12. Claim 16 depends on independent Claim 15. Claims 18-26 depend on independent Claim

17. These dependent claims include all of the features of their respective independent claims. Therefore, these dependent claims should be patentable for at least the reasons that their respective independent claims should be patentable.

Conclusion

Appellant believes that pending Claims 17-25 recite statutory subject matter. Appellant believes that pending Claims 1 and 3-26 are patentable over Gong and Book--Structures. As such, Appellants submit that Claims 1 and 3-26 are patentable over the asserted art.

Appellant respectfully requests that the rejection of Claims 1 and 3-26 be reversed. The Appellant wishes to encourage the Examiner or a member of the Board of Patent Appeals to telephone the Appellant's undersigned representative if it is felt that a telephone conference could expedite prosecution.

Respectfully submitted,
Wagner Blecher LLP

Dated: 09/23/2011

/John P. Wagner, Jr./

John P. Wagner, Jr.
Registration No.: 35,398

Wagner Blecher LLP
Westridge Business Park
123 Westridge Drive
Watsonville, CA 95076

Phone: (408) 377-0500
Facsimile: (831) 722-2350

VIII. Appendix - Clean Copy of Claims on Appeal

1. A method of providing flexible protection in a computer system by decoupling protection from privilege, the method comprising:

enabling receipt of information describing two or more types of protection;

enabling receipt of information describing a relationship between said two or more types of protection and portions of code that are executed in a same privilege level of the computer system, wherein said relationship is not required to be linear and wherein said portions of code are not required to be associated with one or more object oriented classes; and

enabling the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code.

3. The method of Claim 1, wherein said portions of code are domains and each of said types of protection is defined at least in part by one or more domain attributes.

4. The method of Claim 3, wherein said one or more domain attributes includes a domain identifier that specifies to a unique value for a particular domain.

5. The method of Claim 3, wherein said one or more domain attributes includes a Private Key that specifies a unique value for protecting each user that concurrently uses a particular domain.

6. The method of Claim 3, wherein said one or more domain attributes includes a SharedCode Key that specifies a value that a particular domain must use to access code associated with another domain.

7. The method of Claim 3, wherein said one or more domain attributes includes a SharedData Key that specifies a value that a particular domain must use to access data associated with another domain.

8. The method of Claim 3, wherein said one or more domain attributes includes an AllowOthers that specifies a value that a particular domain must use to access code associated with another domain in conjunction with said particular domain performing cross-domain switching to said other domain.

9. The method of Claim 3, wherein said one or more domain attributes includes an AccessOthers Key that specifies a value that is used to request access of code associated with a particular domain on behalf of another domain.

10. A method of providing flexible protection in a computer system by decoupling protection from privilege, the method comprising:

detecting a request from a first portion of code to access a second portion of code, wherein said first and second portions of code are executed in a same privilege level of said computer system and wherein said portions of code are not required to be associated with one or more object oriented classes;

determining whether said first portion of code is allowed to access said second portion of code based on information describing two or more types of protection and also based on information describing a relationship between said two or more types of protection and said portions of code, wherein said relationship is not required to be linear; and

if said relationship specifies that said first portion of code may access said second portion of code, then

allowing said first portion of code to access said second portion of code;

else

not allowing said first portion of code to access said second portion of code.

11. The method of Claim 10, wherein said information describing said two or more types of protection and said information describing said relationships are associated with said portions of code and wherein the method further comprises retrieving said information describing said two or more types of protection and said information describing said relationships .

12. A computer system comprising:

a memory unit; and

a processor coupled to the memory unit, the processor for executing a method for enforcing protection in a computer system by decoupling protection from privilege, the method comprising:

enabling at a user interface receipt of information describing two or more types of protection;

enabling at the user interface receipt of information describing a relationship between said two or more types of protection and portions of code are executed in a same privilege level of the computer system, wherein said relationship is not required to be linear and wherein said portions of code are not required to be associated with one or more object oriented classes; and

enabling at a link-editor the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code.

13. The computer system of Claim 12, wherein said relationship is user definable.

14. The computer system of Claim 12, wherein said portions of code are domains and each of said types of protection is defined at least in part by one or more domain attributes.

15. A computer system comprising:

a memory unit; and

a processor coupled to the memory unit, the processor for executing a method for providing flexible protection in a computer system by decoupling protection from privilege, the method comprising:

detecting at a memory manager a request from a first portion of code to access a second portion of code, wherein said first and second portions of code are executed in a same privilege level of said computer system and wherein said portions of code are not required to be associated with one or more object oriented classes;

determining at said memory manager whether said first portion of code is allowed to access said second portion of code based on information describing two or more types of protection and also based on information describing a relationship between said two or more types of protection and said portions of code, wherein said relationship is not required to be linear; and

if said relationship specifies that said first portion of code may access said second portion of code, then

allowing at said memory manager said first portion of code to access said second portion of code;

else

not allowing at said memory manager said first portion of code to access said second portion of code.

16. The computer system of Claim 15, wherein said information describing said two or more types of protection and said information describing said relationships are associated with said portions of code and wherein the method further comprises retrieving at a loader said information describing said two or more types of protection and said information describing said relationships.

17. A computer-usable medium having computer-readable program code embodied therein for causing a computer system to perform a method of providing flexible protection in a computer system by decoupling protection from privilege, the method comprising:

enabling receipt of information describing two or more types of protection;

enabling receipt of information describing a relationship between said two or more types of protection and portions of code that are executed in a same privilege level of the computer system, wherein said relationship is not required to be linear and wherein said portions of code are not required to be associated with one or more object oriented classes; and

enabling the association of said information describing said two or more types of protection and said information describing said relationship with said portions of code, wherein a first portion of code allowing a second portion of code to access the first portion of code does not depend on the second portion of code allowing the first portion of code to access the second portion of code.

18. The computer-usable medium of Claim 17, wherein said relationship is user definable.

19. The computer-usable medium of Claim 17, wherein said portions of code are domains and each of said types of protection is defined at least in part by one or more domain attributes.

20. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes a domain identifier that specifies to a unique value for a particular domain.

21. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes a Private Key that specifies a unique value for protecting each user that concurrently uses a particular domain.

22. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes a SharedCode Key that specifies a value that a particular domain must use to access code associated with another domain.

23. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes a SharedData Key that specifies a value that a particular domain must use to access data associated with another domain.

24. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes an AllowOthers that specifies a value that a particular domain must use to access code associated with another domain in conjunction with said particular domain performing cross-domain switching to said other domain.

25. The computer-usable medium of Claim 19, wherein said one or more domain attributes includes an AccessOthers Key that specifies a value that is used to request access of code associated with a particular domain on behalf of another domain.

26. The computer system of Claim 15, wherein said second portion of code is allowed to access said first portion of code after a third portion of code accesses said second portion of code and wherein said third portion of code is not required to allow access to said first portion of code.

IX. Evidence Appendix

No evidence is herein appended.

X. Related Proceedings Appendix

No related proceedings.